НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри

_____ О.В.Коваль
(підпис) (ініціали, прізвище)

“___”_____2019 р.

ДИПЛОМНА РОБОТА
на здобуття ступеня бакалавра

з напряму підготовки
6.050103 “Програмна інженерія”

на тему: Розробка серверної частини для системи обліку успішності студента

Виконав: студент 4 курсу, групи ТІ-51

Іккес Давид Павлович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. в. Матях Сергій Володимирович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
(підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050103 “Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту
Іккесу Давиду Павловичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Розробка серверної частини для системи обліку успішності студента ”
керівник роботи ст. в. Матях Сергій Володимирович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від 22 03 2019р. № 1325-с

2. Строк подання студентом роботи ___ _____ 201__ р.
3. Вихідні дані до роботи Python проект
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та засоби для обліку успішності студента, спроектувати архітектуру системи обліку успішності студента. розробити програмне забезпечення, розробити API інтерфейс
5. Перелік ілюстраційного матеріалу
1. Вступ 2. Актуальність 3. Мета та завдання роботи 4. Існуючі програмні рішення 5. Використані програмні засоби 6. Архітектура програмного комплексу 7. Опис функціональності 8. Додаткові модулі 9. Висновки

Дата видачі завдання ”___”_____ 201__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2.	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	Захист програмного продукту		
7.	Оформлення пояснювальної записки		
8.	Передзахист		
9.	Захист		

Студент _____ Іккес Д.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матях С. В.
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Будь-який освітній процес супроводжується оцінкою отриманих знань для того, щоб коригувати процес та покращувати ефективність засвоєння навчального матеріалу студентами.

До моменту створення технічних засобів для роботи з електронною інформацією, використовувались паперові ресурси: щоденники, журнали успішності та інше. Технічні засоби полегшують процес запису, пошуку та обробки даних. Інформацію про навчальний розклад, предмети, студентів та викладачів, можна легко поширювати та модифікувати, зміни моментально доступні іншим. Це спрощує багато аспектів у яких виникали проблеми раніше, проте повністю всіх проблем це не вирішує.

Якщо потрібно оцінювати процес за весь семестр чи рік, виконання запису проміжних результатів полегшить процес визначення кінцевого результату роботи студента. Від типу записів залежать і методи, які будуть використані для формування остаточної оцінки. Вимоги до оцінювання в більшості випадків не очевидні для студентів, мають відмінності у різних викладачів та можуть змінюватись протягом семестру, що лише погіршує ситуацію з погляду студентів.

Для вирішення цієї проблеми розроблено систему, яка дозволить поширювати цю інформацію між студентами та дати можливість задавати питання пов’язані з навчальним процесом. Питання з відповідями, які розподілені між навчальними групами чи спеціалізаціями, полегшить пошук необхідної інформації та швидкість осягнення загального контексту предметної області.

Питання завжди виникають в процесі навчання, тому доволі ефективним буде оцінка кількості та якості заданих питань. Оцінку можуть ставити інші користувачі, на основі цього можна сформувати рейтинг успішності та певної статистичної оцінки засвоєння матеріалу.

При участі викладачів, які зацікавлені в поширенні знань, та можливості задавати питання анонімно, можна виключити фактор особистого відношення та

загалом покращити процес спілкування в площині студент-викладач. Також це буде означати залучення експертів, які мають глибокі знання в предметній області, що дуже важливо для навчального процесу.

Було запропоновано розробити модуль для доступу до розкладу університету, що дозволяє отримувати необхідну інформацію про предмети, викладачів і сам розклад. Дана частина системи дозволить підтримувати актуальність поточної навчальної інформації, а також виключити ряд проблем, які з цим зв’язані.

Іншим компонентом, який повинна включати система виступає сервіс з функціональністю для роботи з основними сутностями. Даний сервіс має оперувати питаннями, відповідями та користувацькою інформацією та виступати в якості елемента, який забезпечує контроль основного потоку виконання.

1. ЗАДАЧА РОЗРОБКИ СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ ОБЛІКУ УСПІШНОСТІ СТУДЕНТА

Метою роботи є розробка системи для покращення ефективності навчального процесу з акцентом на обміні актуальною інформацією, створення рейтингу на основі запитань та відповідей, а також забезпечення поділу всієї інформації за категоріями.

Для вирішення проблем, які існують у навчальному процесі в вищих навчальних закладах потрібна масштабна платформа. Для впровадження і підтримки цього потрібні значні ресурси, тому в ході дослідження було визначено основні проблеми, які можна вирішити з частковим використанням інших сервісів. Як результат потрібно створити застосунок, з мінімальними втратами та максимальним покриттям можливих вирішених проблем.

Таким чином як основну, запропоновано таку функціональність:

1. Інструменти для комунікації між студентами та викладачами;
2. Запитання-відповіді, для полегшення процесу пошуку необхідної інформації при навчанні;
3. Можливість переглядати наявні дисципліни, доступну для них інформацію та час, коли проходять заняття.

Система повинна забезпечувати зручність у використанні як для студентів, так і для розробників, які будуть займатись підтримкою та подальшою реалізацією інших компонентів системи обліку успішності.

Система повинна забезпечувати потрібний рівень швидкодії між внутрішніми компонентами та в результаті для кінцевого користувача.

Облік успішності студента є однією з частин навчального процесу, що дозволяє належним чином контролювати процес. Це також допомагає визначати проблеми, які виникають у студентів протягом навчання, і вчасно на них реагувати

чи змінювати навчальні програми. Також облік успішності можна назвати елементом зворотного зв’язку між студентом та викладачем.

1.1 Педагогічний підхід до організації обліку успішності студента

Інформація, яка передається каналами зворотного зв’язку, у відповідності до теорії управління, повинна відповідати визначеному набору вимог:

1. повнота — необхідні для побудови висновків об’єм і зміст інформації, що допомагає здійснювати управлінські рішення;
2. об’єктивність — відповідність між отриманою інформацією і станом підсистеми (до прикладу, рівень досягнень студентів у навчанні);
3. актуальність та оперативність — своєчасний доступ до інформації для подальшого аналізу і оперативного прийняття рішень за необхідністю і характером зауважень викладача;
4. неперервність у часі — постійне постачання інформації в навчальному процесі для підтримання стану підсистеми і можливості постійного управління[1].

При навчанні у студента повинні формуватись певні вміння та навички,творчі здібності та зацікавленість в отриманні нової інформації, іншими словами жага до знань. Контроль знань студентів повинен охоплювати всі розділи навчальної програми і носити всесторонній характер.

Для того, щоб навчальний процес був ефективним та забезпечував належну якість потрібно, щоб він виконував ряд функцій:

1. виховна функція — надає студентам можливість здобути навички адекватної оцінки власних досягнень та своєї успішності. Також ця функція дозволяє формувати власну відповідальність та краще організовувати своє особисте навчання;
2. мотиваційна функція — забезпечує зацікавленість студента в отриманні

нових знань, стимулює активність в навчальній та науковій діяльності;

3. коригувальна функція — дозволяє виявляти проблемні місця у знаннях студентів, допомагати знайти оптимальний шлях в досягненні поставлених цілей;
4. методична функція — дозволяє викладачу оцінювати методи, які використовуються і вносити оперативні зміни в навчальний процес;
5. дидактична функція — забезпечує не тільки систематизацію, а і узагальнення всього матеріалу по дисципліні, яка вивчається;
6. перевірочна функція — забезпечує оцінку рівня засвоєння навчального матеріалу, здобутих знань та навичок[2, 3].

Умовою для реалізації функцій контролю результатів навчання являється використання різних форм оцінки прогресу студента в навчанні:

1. попередній контроль спрямований на визначення початкового рівня студента, як результату засвоєного матеріалу на попередніх етапах навчання;
2. поточний контроль дозволяє оцінювати діяльність студента впродовж навчального процесу, представляє реальну оцінку засвоєння матеріалу;
3. проміжний контроль складає перевірку засвоєного студентом матеріалу в певній точці, по закінченню теми, розділу чи логічного завершеної частини дисципліни;
4. підсумковий контроль проводиться для оцінки результатів навчання, як правило, виконується в кінці семестру в формі заліку чи екзамену і завершує вивчення дисципліни.

Якісну роботу описаних функцій може забезпечити хороший зворотній зв’язок між студентом та викладачем і являється необхідною умовою для побудови навчального процесу.

Для визначення якості засвоєного студентом матеріалу та навчальної програми загалом, викладач повинен сформулювати основні вимоги та критерії оцінки успішності навчання. Вони повинні показувати ступінь здібностей студента до виконання визначених видів робіт в навчальній та професійній діяльності. Успішність являється латентним, тобто прихованим, параметром, адже у викладача

немає можливості напряму вимірювати знання студента за вибраною навчальною дисципліною. Проте викладач має можливість провести оцінку знань за попередньо вибраними параметрами, що дадуть можливість побудувати характеристику для оцінки знань. Тут виникає питання в об’єктивності цієї оцінки та ступені валідності цих параметрів і методів для вимірювання рівня успішності навчання.

Для побудови моделі оцінки необхідно вирішити такі проблеми:

1. вибір та обґрунтованість інформативних параметрів;

2. вибір схеми оцінювання;

3. вибір алгоритму побудови підсумкової оцінки.

Від вибраного алгоритму оцінки напряму залежить ступінь мотивації студентів, який і визначає підсумковий бал з дисципліни.

1.2 Основні методи підсумкового контролю

Для відповідності сучасним вимогам до системи управління та контролю якістю освіти потрібно забезпечувати належний рівень об’єктивності оцінки знань. Виключити фактор суб’єктивної оцінки не так вже і просто, для того, щоб розуміти підводні камені цієї проблеми можна проаналізувати основні методи підсумкового контролю. Саме підсумковий контроль дозволяє в повній мірі побачити проблеми, які притаманні для обліку успішності впродовж навчальних семестрів.

Підсумкова оцінка напряму залежить від якості поточної роботи впродовж семестру, також існує велика кількість математичних моделей для обліку поточної успішності. Від того як буде обрана система оцінювання залежить рівень мотивації студента впродовж періоду викладання дисципліни, як результат від цього будуть залежати і кінцеві результати навчання [4].

Так як прослідкується пряма залежність між результатами оцінювання студентської роботи в кінці семестру та в період від початку до кінця, було б доцільним розглянути методи підсумкового контролю.

1.2.1 Підсумкова бальна екзаменаційна оцінка

Іншим методом підсумкового контролю використовується оцінка отримана в результаті екзамену. В сучасній освітній системі вищої школи використовується шкала, яка має чотири категорії від оцінки “незадовільно”, що відповідає оцінці 2, до “відмінно”, що у свою чергу, відповідає оцінці 5. При використанні даного підходу можна розуміти усне опитування по наперед підготовлених білетах і зазвичай проводиться по закінченню навчального предмета. Сама оцінка є доволі суб’єктивною, адже базується на суб’єктивній думці викладача про студента, який буде відповідати на вибраний білет.

Суттєві недоліки, які тягне за собою використання даного підходу при оцінці знань це:

1. протягом семестру враховується недостатньо для формування повної оцінки знань та прикладених до навчання сил;

2. оцінка є доволі не об’єктивною, адже має місце відношення викладача до студента;

3. обрані білети не дозволяють повністю покрити матеріал, який вивчався протягом семестру.

Велика кількість досліджень, які було проведено доводять, що така система оцінювання являється доволі застарілою і потребує покращення для подальшого застосування [2, 3, 4].

1.2.2 Параметричні моделі Раша і Бірнбаума

У 80-х рр. XX століття в педагогічних методах оцінки успішності студентів широкого поширення набули методи сучасної західної теорії тестування Item Response Theory (IRT) [5].

Математичні моделі IRT використовуються при оцінці ймовірності правильної відповіді тестованого на завдання різного рівня складності. В основі теорії IRT лежить припущення про те, що навчальні досягнення учня в певній галузі знань можна оцінювати за допомогою латентного (прихованого) параметра, званого рівнем

підготовленості, в той час як для характеристики кожного завдання, що вирішується учням, використовується інший латентний параметр, званий рівнем складності завдання. Після проходження учнями тестування, значення даних латентних параметрів оцінюються із застосуванням методів статистичного аналізу. Слід зазначити, що тестованих має бути достатньо багато, оскільки мова йде про статистичний аналіз. Для обчислення статистичних оцінок прихованих параметрів задається математична модель, яка характеризує залежність ймовірності рішення окремого завдання конкретним учням (ймовірності успіху) від рівня підготовленості випробуваного і складності завдання (функцією успіху). У якості функцій успіху на практиці часто застосовують однопараметричну модель Раша і двопараметричну модель Бірнбаума. В Україні модель Раша частково використовується при аналізі результатів ЗНО. На відміну від класичної теорії тестування, в IRT індивідуальний бал учня за підсумками тестування не залежить від складу сукупності тестованих, що змінюється числа завдань і їх складності, що є її головною перевагою.

Крім цього, сучасна теорія тестування має наступні переваги:

1. висока точність вимірювань за рахунок більшої градації оцінки;

2. об’єктивність результатів, тобто виключається вплив суб'єктивних факторів;

3. можливість включення в тести матеріалу, що охоплює всі теми вивченого матеріалу навчальної дисципліни.

Однак у практичній діяльності викладачів вузів дані моделі не застосовуються, оскільки з одного боку для їх використання потрібно збирати доволі великі вибірки (тисячі осіб), чого не дозволяє зробити розмір академічної групи, а з іншого громіздкість математичних обчислень.

1.2.3 Тестування

Як педагогічний інструмент для вимірювання рівня знань використання тестування є хорошим і практичним підходом. Набір завдань поділених за різними рівнями складності дозволяє якісно оцінити структуру та виміряти рівень знань і навичок, набутих студентами. Підсумкове тестування використовується на

заключному етапі вивчення дисципліни для того, щоб забезпечити об'єктивну оцінку результатів навчання, зосереджуючись на природі змісту навчання в навчальному матеріалі.

Заключний тест повинен охоплювати досить великий обсяг пройденого матеріалу, включаючи завдання, спрямовані на оцінку вже існуючих навичок та знань. Завдання, що містяться в тесті, повинні бути однозначними, максимально короткими і чітко сформульованими.

Як правило, тест складається з двох частин: інструкцій і тестових об'єктів. Посібник містить інструкції для тестової структури, представленої суб'єктам, час, виділений для завершення, те, як зберігаються правильні відповіді тощо. Нумеровані завдання тестування організовані після інструкції в порядку збільшення складності. Це робиться для того, щоб дозволити студентам з поганою готовністю виконувати певну кількість завдань, в іншому випадку, якщо почати тестування зі складних завдань, то є вірогідність не впоратися зі складними завданнями і витратити багато часу без виконання простих завдань.

Переваги тестування порівняно з традиційними формами контролю включають:

1. підвищення об'єктивності контролю за рахунок вилучення зовнішніх факторів;

2. присутня можливість охопити весь обсяг пройденого матеріалу;

3. широкі можливості для автоматизації.

Проте тестування як метод контролю не враховує роботу студента протягом семестру. Також присутній елемент випадковості, коли не підготовлений студент відповідає правильно за рахунок вибору випадкових відповідей.

Тестування можна застосовувати і для поточного контролю роботи студента протягом семестру.

1.2.4 ECTS

В останні роки мобільність студентів стає все більш важливою у навчальному

процесі університету. Це одна з головних складових інтеграційних стратегій Європейського простору вищої освіти. Академічна мобільність забезпечує підтримку спільних освітніх програм і навчальних програм, розробку загальних методів оцінки якості освіти та обміну освітою.

Студентська академічна мобільність означає можливість переходу студента в іншу освітню організацію (у своїй країні або за кордоном), після чого студент повертається до свого початкового навчального закладу. Слід зазначити, що, крім перерахунку загальної кількості трудових витрат студентів з метою набуття змісту дисципліни, ця система також повинна включати можливість оцінки якості набуття знань і навичок, набутих студентом у цій дисципліні.

Сьогодні така система оцінювання набула неабиякої популярності та у багатьох країнах використовується під назвою Європейська система переказу та накопичення кредитів (ECTS) - в центрі знаходиться діяльність студентів при вивченні певної дисципліни та при формування навчальних програм. Система ECTS діє як невід'ємна частина впровадження Болонського процесу і дозволяє академічну та трудову мобільність як для студентів, так і для випускників.

Питання переоцінки якості набутих знань і навичок продовжує залишатися предметом досліджень і дебатів у європейських освітніх колах. Шкала оцінювання, розроблена університетом однієї країни, потрібно представити подібно до шкали оцінювання, прийнятого в іншій країні, це питання є досить складним і неясним.

Система ECTS передбачає, що продуктивність студента розраховується відповідно до прийнятої національної шкали, тоді як додаткову шкалу доцільно використовувати шкалу статистичної класифікації ECTS grading scale запропоновану Європейською Комісією в середині 1990-х років як частину ECTS в цілому.

Ось основні положення системи ECTS:

1. при розрахунку кредитів кількість трудозатрат враховує будь-яку студентську роботу: робота в класі, самостійна робота, участь у семінарах, підготовка звітів, лабораторна робота, написання реферату, курсова робота тощо;
2. відвідування студентом занять не впливає на розмір нарахованих кредитів,

але адміністрація університету або викладач можуть призначати скорочення кількості вже набраних кредитів у разі, якщо студент не відвідує аудиторні заняття.

3. трудозатрати студентів необхідні для досягнення очікуваних результатів навчання, оцінюються в кредитах ECTS (кредити ECTS); один навчальний рік має 60 кредитів ECTS, що становить близько 1500-1800 академічних годин, тому один кредитний бал становить 25-30 академічних годин;
4. для отримання ступеня бакалавра, студент повинен заробити від 180 до 240 кредитів ECTS (залежно від періоду навчання), а для отримання ступеня магістра щонайменше 300;
5. кількість кредитів має бути цілим числом (як виняток можна додати 0,5 кредиту);
6. кредити нараховуються після завершення навчальної програми в цілому або як окремий структурний елемент навчального плану та успішної оцінки досягнутих результатів навчання;
7. оцінка не впливає на суму накопиченого кредиту;

Оцінка в системі оцінювання ECTS grading scale проводиться в кілька етапів. По-перше, встановлюється мінімальне значення, що вказує на успішність засвоєння дисципліни та отримання кредиту, потім, на основі суми балів, студенти поділяються на дві підгрупи відповідно до наступних умови: студенти, у яких сума балів була не менше встановленої критичної величини отримують “залік”; оцінку “незалік” отримують студенти, які набрали менше критичного значення, іншими словами, кредитування цих студентів не присвоюється.

Студенти, які отримують оцінку “залік”, у свою чергу поділяються ще на кілька підгруп.

Градації по шкалі ECTS grading scale визначають згідно статистичного розподілу, тому в кінці вивчення дисципліни вони розміщуються в порядку спадання успіху.

За місцем, де студент брав участь у рейтингу, йому надається одна з наступних позитивних балів за шкалою ECTS:

1. оцінка A (10% від загальної кількості успішно пройшли) є видатним результатом з незначними помилками;
2. оцінка B (25%) - вище середнього, але з деякими помилками;
3. оцінка C (30%) - загалом хороший рівень, але є ряд недоліків;
4. оцінка D (25%) - відбуваються значні помилки або недоліки;
5. оцінка E (10%) - задовольняє мінімальне падіння вимог.

Для студентів, які отримали "не кредитуються" встановлюються наступні цитати:

1. оцінка FX - вимагає додаткового вивчення матеріалу;
2. оцінка F - вимагає значних додаткових робіт.

Шкала ECTS базується на статистичному підході, тому варто зазначити, що мінімальної кількості студентів, необхідних у групі, достатньо для поділу - 30 осіб (хоча більша кількість студентів краще) .

Сьогодні ECTS успішно використовується у всьому світі. Основними перевагами системи є:

1. прозорість в оцінці результатів освітньої діяльності;
2. видимість у визначенні фактичного навантаження студента при виконанні різних освітніх активностей;
3. сумісність навчальних планів університетів різних країн через їх структурування;
4. перехід вітчизняних університетів до системи шкали оцінювання ECTS;
5. статистичний підхід, що лежить в основі системи ECTS, вимагає від усіх учнів сумлінного ставлення до навчання та прагнення досягти максимальних результатів у освітній діяльності, щоб продовжити хорошу роботу, яка, на жаль, відсутня в нашій дійсності; з точки зору побудови рейтингової шкали, це означає, що вона не може бути побудована на основі статистичного розбиття;

Якщо фінальна форма управління встановлює тест, який може виконувати студент, то ми можемо визначити лише кількість зароблених кредитів, але не якість їхнього володіння. Для статистичного розподілу потрібно велике чисто студентів, що

в свою чергу неможливо для вітчизняних ВНЗ, які часто мають невелику академічну групу.

На даному етапі впровадження ECTS grading scale викликає труднощі, оскільки для цього потрібно проводити велику кількість реформ, перепоною для яких буде зміна структури оцінювання.

Проте така оцінка, паралельно з методами оцінки, що використовуються у вітчизняних вузах буває доволі корисною адже забезпечує більшу гнучкість у порівнянні зі стандартними методами оцінки, а також покриває поточний контроль, як частину загальної оцінки за пройдену дисципліну.

1.3 Висновки до розділу

Була поставлена задача і проаналізовано можливі проблеми при створенні продукту. Для подальшого проектування і розробки веб-додатку виділено основні цілі та умови, що мають бути виконані для того, щоб кінцевий користувач отримав всі необхідні функції.

2. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ДЛЯ ОБЛІКУ УСПІШНОСТІ

З технологічним прогресом зростають і вимоги до програмного забезпечення, яке супроводжує всі процеси. Автоматизація цих процесів дозволяє покращувати рівень розвитку суспільства загалом, однією з таких сфер є освітня.

Сьогодні можна зустріти багато реалізацій систем для обліку успішності студента. Всі вони в значній мірі вирішують проблеми, які виникають при розробці такого роду систем, проте з цього випливають і проблеми, які роблять їх надто складними або не зручними у використанні.

Було вирішено розробити систему, яка хоча б частково вирішує основні проблеми в існуючих системах.

2.1 Система “Електронний кампус НТУУ КПІ”

Це програмне забезпечення представлене як платформа, що призначена для обліку успішності студентів у НТУУ “КПІ ім.Ігоря Сікорського”, використовується для підтримки студентів та інших робітників університету протягом навчального процесу [1].

Функції, які повинен виконувати ресурс:

1. забезпечення зворотного зв’язку у площині студент-викладач;
2. забезпечення актуальною інформацією, щодо навчального процесу, поточних подій;
3. формування єдиного інформаційного ресурсу для університету.

За результатами опитування студентів, було виявлено, що використовують його лише 13% опитаних, що є досить низьким показником. Це може свідчити лише про те, що при розрозробці не були враховані думки тих, хто формує кінцеву вибірку

користувачів.

Першим основним недоліком цієї системи є застарілий інтерфейс, приклад продемонстровано на рисунку 2.1. Зовнішній вигляд для будь-якого ресурсу, якому притаманний соціальний аспект, дуже важливий, адже це перше з цим знайомиться користувач.

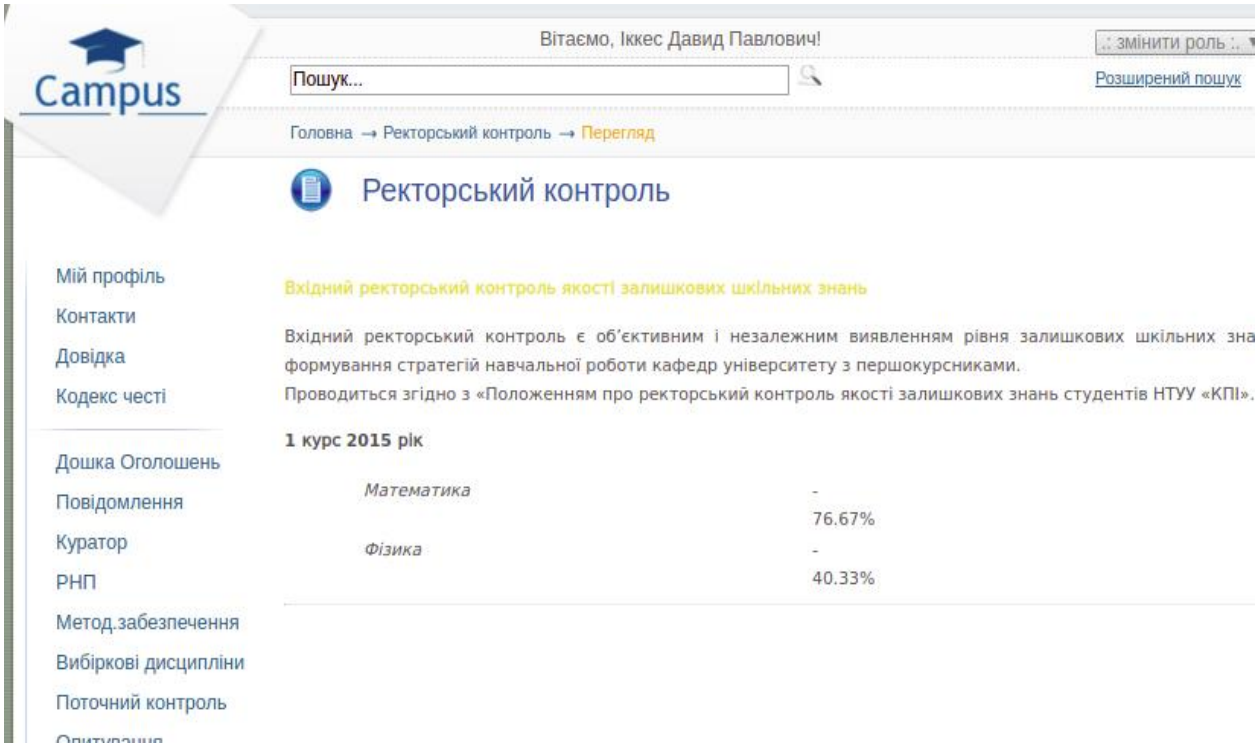


Рисунок 2.1 — Приклад користувацького інтерфейсу Campus

Якщо ресурс спрямований на використання впродовж доволі довгого терміну та повинен забезпечувати зручний доступ до великих об'ємів інформації, то ця інформація повинна бути подана в зручному для користувача вигляді.

В цьому випадку інтерфейс відіграє дуже важливу роль. Правильно побудований інтерфейс ресурсу дозволить забезпечити лояльність серед користувачів та уникнути відтоку вже існуючих на даний момент користувачів.

Другим основним недоліком є відсутність актуальної інформації, яка є невід'ємною частиною навчального процесу. Для того, щоб забезпечувати належний рівень фахівців, яких готує навчальний заклад, потрібно забезпечувати студентів

актуальною інформацією впродовж всього навчального процесу.

Виникають протиріччя між описаним функціоналом та наявною на ресурсі інформацією. Приклади невідповідності можна побачити на рисунку 2.2, де інформація відстає від поточної на кілька років, що є недопустимим для такого роду ресурсів.

Рисунок 2.2 — Дошка оголошень на ресурсі Campus

Відсутність актуальної інформації напряму впливає на якість будь-яких

освітніх процесів, що проходять в навчальних закладах, тому цей недолік, можливо, є одним з головних, які притаманні цьому ресурсу.

2.2 Загальний огляд існуючих LMS

Облік успішності студента включає роботу з ресурсами, які потрібні для вивчення предметної області, що відноситься до поточної дисципліни. Матеріали повинні бути оформлені зрозумілим для студента та викладача чином, аби робота з ними була зручною в навчальному процесі. Для того, щоб зберігати результати поточного та підсумкового контролю та остаточного формування навчального рейтингу серед студентів було б доцільним мати потрібні інструменти, які забезпечують потрібний функціонал. На думку приходить клас програмного забезпечення під назвою LMS. Такий тип додатків забезпечує велику кількість ресурсів, які нададуть можливість спростити процес контролю та оцінки впродовж навчання. Здебільшого це вільне програмне забезпечення, яке спрямоване саме на полегшення контролю та ефективності навчання.

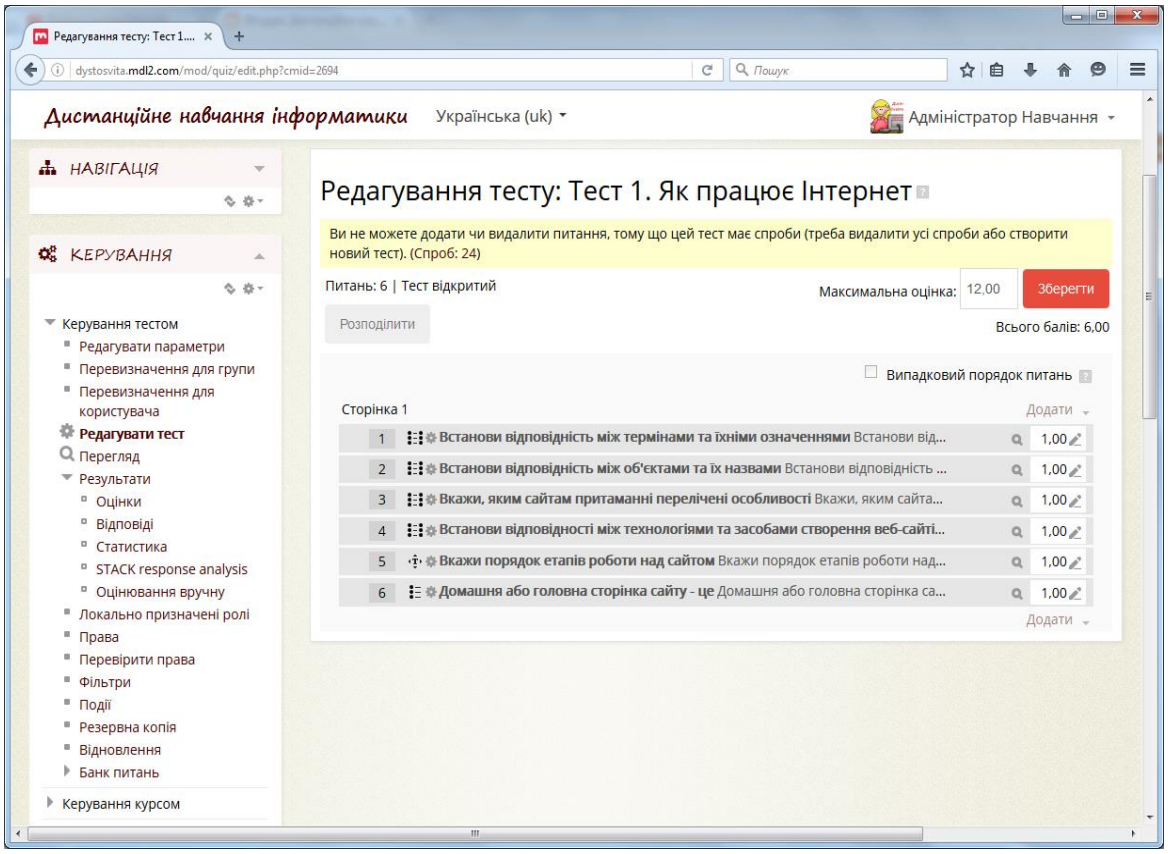


Рисунок 2.3 — Інтерфейс дистанційного курсу Moodle

Такого роду системи застосовуються здебільшого для дистанційного навчання, проте знайшли застосування і в рамках освіти у вищій школі. Проте повноцінне застосування вимагає затрати значних ресурсів на освоєння існуючого функціоналу та створення матеріалів для наповнення системи контентом.

Як приклад, такої системи можна розглянути платформу Moodle (рисунок 2.3), яка є доволі поширеним прикладом ресурсу для управління навчанням у безкоштовному сегменті програмного забезпечення.

2.3 Висновки до розділу

Було розглянуто існуючі рішення, що дозволяють вирішити проблеми, які існують при обліку успішності студента. В них виявлено ряд недоліків, які перешкоджають повноцінному впровадженню такого роду систем в навчальний процес.

3. ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

В ході дослідження проблеми та методів її вирішення, було прийняте рішення розробляти прогрмний комплекс на основі мови програмування Python та фреймворк Flask. В якості середовища розробки використовується PyCharm.

3.1 Середовище розробки PyCharm

Багатофункціональний редактор коду для мови Python під назвою PyCharm створений для швидкої та ефективної розробки програмного забезпечення. Наявна велика кількість інструментів, які дозволять сконцентруватись на написанні саме бізнес-логіки застосунку не відволікаючись на рутинні речі: налаштування віртуального середовища, підключення системи контролю версій, встановлення зовнішніх бібліотек тощо.

Для розробки даної системи було використано останню версію редактору — 1.34. Серед особливостей цього редактору коду можна відмітити такі пункти:

1. безкоштовний текстовий редактор з відкритим вихідним кодом.
2. файлова система проекту для зручного переходу між компонентами;
3. широкі можливості для тестування та зневадження;
4. підтримка зовнішніх ресурсів для впровадження програмного продукту;
5. підтримуються основні VCS та доступні інструменти, що полегшують роботу з цими системами;
6. можливість розробки та встановлення власних плагінів для розширення функціоналу.

7. підтримка розумного вводу IntelliSense;
8. велика база доступних розширень та плагінів, що зроблять вашу розробку ще швидшою;
9. підтримка основних десктопних платформ;
10. широкий вибір користувацьких тем для інтерфейсу.

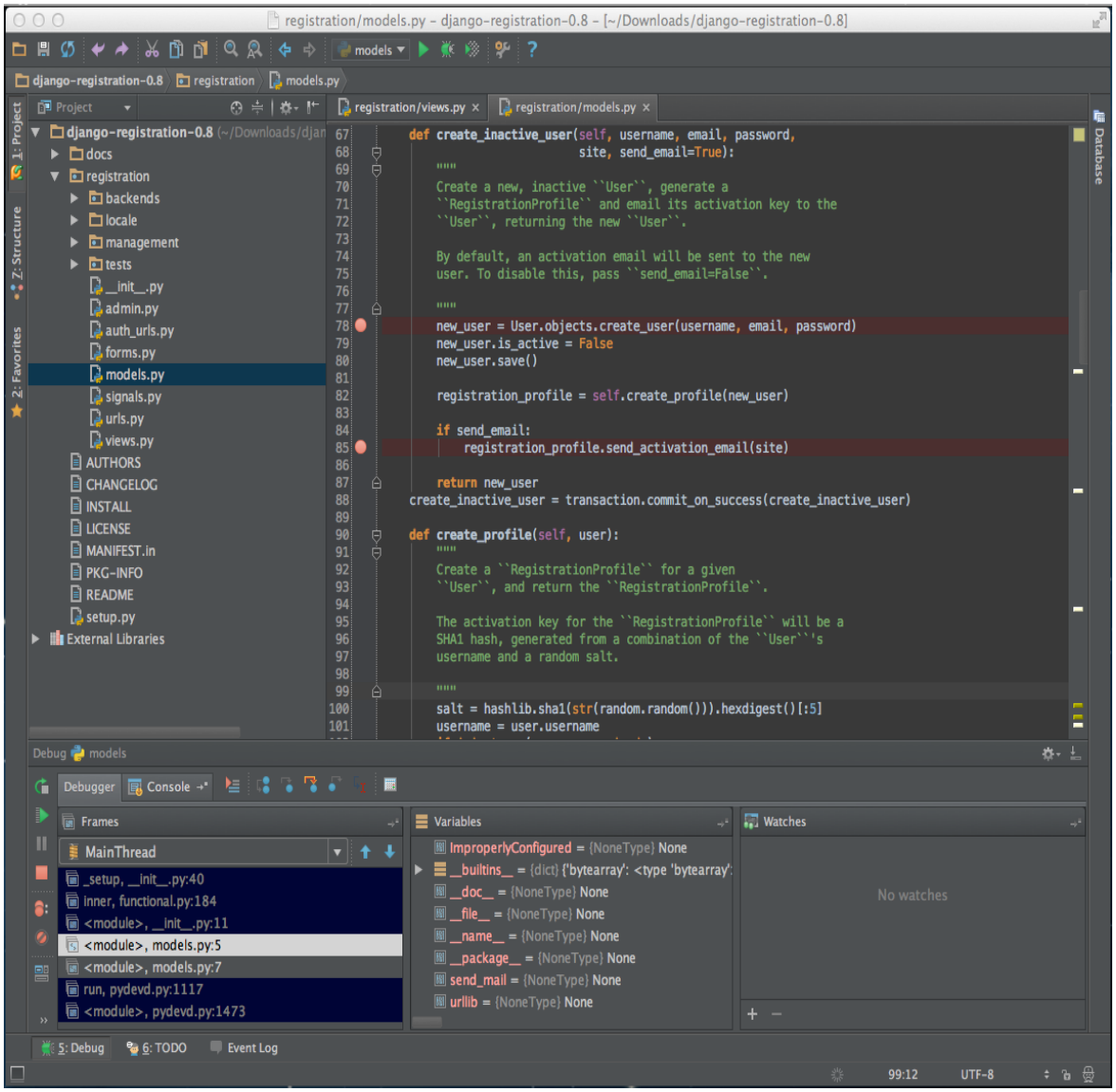


Рисунок 3.1 — Процес роботи зневаджувача у PyCharm

Розробка, яка відбувається в реальному часі потребує швидкого виправлення помилок, які будуть зустрічатись в коді, допомагають стандартні інструменти цього середовища розробки для визначення помилок, тестування і рефакторингу, що зображено на рисунку 3.1.

3.2 Мова програмування

Мова Python — це інтерпретована високорівнева мова програмування, яка підтримує імперативну, об’єктно-орієнтовану та функціональну парадигми. Використовується строга динамічна типізація та виділення робочих блоків коду за допомогою відступів. Це робить з Python ідеальний інструмент для швидкого написання коду, в цьому також допомагає високий рівень читабельності і сприйняття тексту програми в порівнянні з С-подібними мовами програмування.

Основні переваги цієї мови програмування:

1. Стандартний дистрибутив містить модулі, які покривають 90% задач, існують готові дистрибутиви для вирішення потрібних задач;
2. Може бути розширена функціями та типами даних написаних на C/C++;
3. Відкритий сирцевий код;
4. Використовується для розробки систем на базі штучного інтелекту, тому має широкі можливості по обробці даних;
5. Зрозумілий та лаконічний синтаксис;
6. Висока швидкість розробки програм.

```
import MySQLdb

db = MySQLdb.connect(host="194.61.183.124",
                     user="bergenson",
                     passwd="qwerty123",
                     db="mi6db")
cur = db.cursor()
cur.execute("SELECT * FROM SECRET_AGENTS_LIST")

for row in cur.fetchall():
    print(row[0])

db.close()
```

Рисунок 3.2 — Процес роботи зневаджувача у PyCharm

Було обрано саме цю мову програмування, з прикладом коду можна ознайомитись на рисунку 3.1, адже швидкість написання коду дозволяє більше часу виділити на проектування перед цим, а також на тестування після написання коду. Також ця мова програмування має багаті функціональні можливості, що може значно покращити якість вихідного коду.

3.3 Модулі для серверної частини

Фреймворк Flask - який забезпечує мінімальний функціонал для створення простих та масштабованих веб-додатків.

Даний фреймворк побудований на базі модуля Werkzeug, який виконує роль маршрутизатора та включає в себе реалізацію повнофункціональних об’єктів для створення запитів та відповідей для взаємодії з мережею через протокол HTTP. До стандартного набору включено шаблонизатор Jinja2, який дозволяє вбудовувати в гіпертекстову розмітку змінні та цілі вирази для викликів з Python.

Також Flask містить влаштований дебагер та сервер для розробки. Саме мінімальний набір модулів, які складають ядро модуля відкриває широкі можливості для масштабування, адже зовнішні пакети оновлюються частіше, аніж ядро фреймворку.

Цей фреймворк використовується як основа для розробленого сервісу, що буде виконувати роль контролера основного потоку виконання програмної реалізації системи.

3.3.1 Flask-Restful

Цей додатковий модуль використовується як розширення в якому реалізовано обгортку для стандартного Flask спрямованого на розробку саме REST API інтерфейсів.

Використання такого підходу дає ряд переваг:

1. доступні класи дозволяють імплементувати ресурси з базовими HTTP-методами для використання однотипних запитів в одному місці;

2. можливість використовувати наперед визначені класи для збору аргументів, які приходять у запитах;

3. спрощена процедура перетворення запитів у JSON-об’єкти.

Це дає змогу зосередитись на реалізації основної бізнес-логіки сервісу на рівні потрібних абстракцій та не заглиблюватись в нижчу за рівнем абстракції реалізацію модулів.

Методи, які представлені в якості ресурсів відповідають за обробку HTTP-запитів, що надходять до серверу. Як єдиний інтерфейс для спілкування через мережу використовується протокол під назвою HTTP (HyperText Transfer Protocol). Цей протокол визначає правила за якими спілкуються різні програмні засоби в мережі використовуючи низькорівневі протоколи для безпосередньої передачі даних через мережу. Методи, які використано при розробці програмного продукту включають в себе:

— GET — використовується для отримання даних, що розташовані на веб-сервері, передаючи параметри та заголовки;

— POST — використовується для створення нових даних для розташування на веб-сервері, містить тіло запиту та заголовки;

— PUT — використовується для оновлення вже існуючих на веб-сервері даних, аналогічно до попереднього, передає тіло запиту та заголовки;

— DELETE — видалення існуючих на веб-сервері даних.

Протокол HTTP найчастіше використовується для того, щоб реалізувати будь-які веб-сервіси, з єдиним визначеним інтерфейсом для передачі даних через мережу.

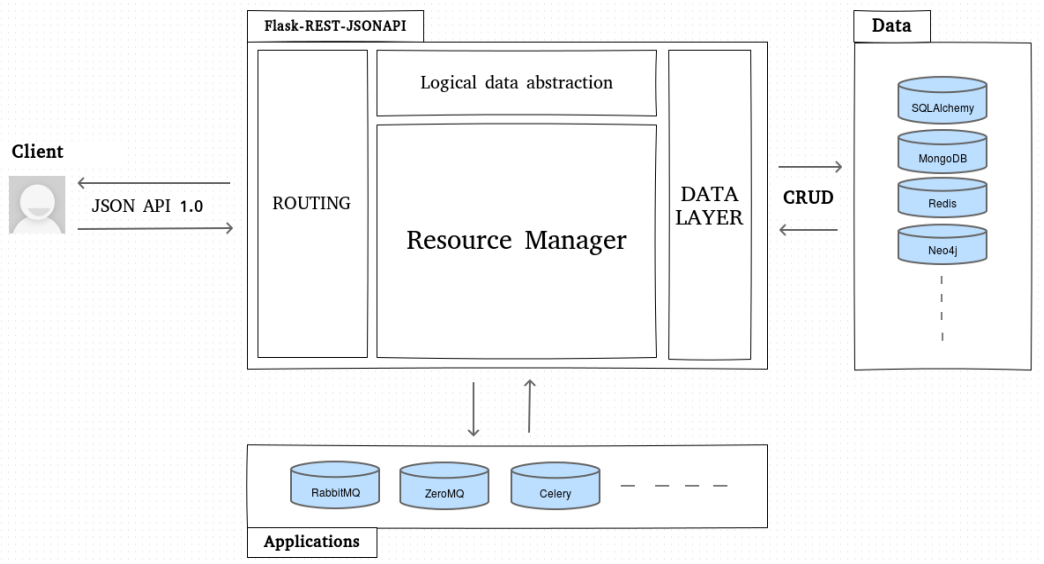


Рисунок 3.3 — Процес роботи зневаджувача у PyCharm

Базова структура додатку, що використовує Flask-RESTful зображено на рисунку 3.3. У більшості випадків такі додатки є частиною архітектурного шаблону MVC і виконують роль Контролера (Controller), що об’єднує всі інші компоненти, виконуючи відображення у Представленні (View) та використовуючи дані з Моделі (Model).

3.3.2 Flask-JWT-Extended

Використовується стандарт для аутентифікації користувача за допомогою токенів доступу, які зазвичай передаються у заголовках. А у свою чергу сам токен теж має свою структуру, яка представлена у вигляді заголовка, вмісту та підпису. Передається у вигляді заголовку в полі Authorization як Bearer-Token:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

Забезпечує більш надійний доступ до ресурсів сервісу та забезпечує надійність зберігання даних користувача.

3.3.3 MongoDB

Раніше всі базові потреби в зберіганні інформації можна було забезпечити за допомогою простого бінарного або текстового файлу, але зараз зросли вимоги до

програмного забезпечення та методів зберігання інформації від яких залежить якість взаємодії між компонентами системи та з кінцевим користувачем.

Модель бази даних під назвою NoSQL використовуються в основному для збереження великих кількостей неструктурованих даних, проте не забезпечують механізму транзакцій, через що трішки втрачають в надійності. В обраній базі даних дані зберігаються як документи JSON.

Стандарт JSON - це представлення даних як об’єкта на мові JavaScript, що дозволило стати саме цей тип надзвичайно універсальним і стати де-факто стандартом при роботі з файлами через мережу.

Для взаємодії з базою даних вирішено використовувати хмарний сервіс Atlas MongoDB.

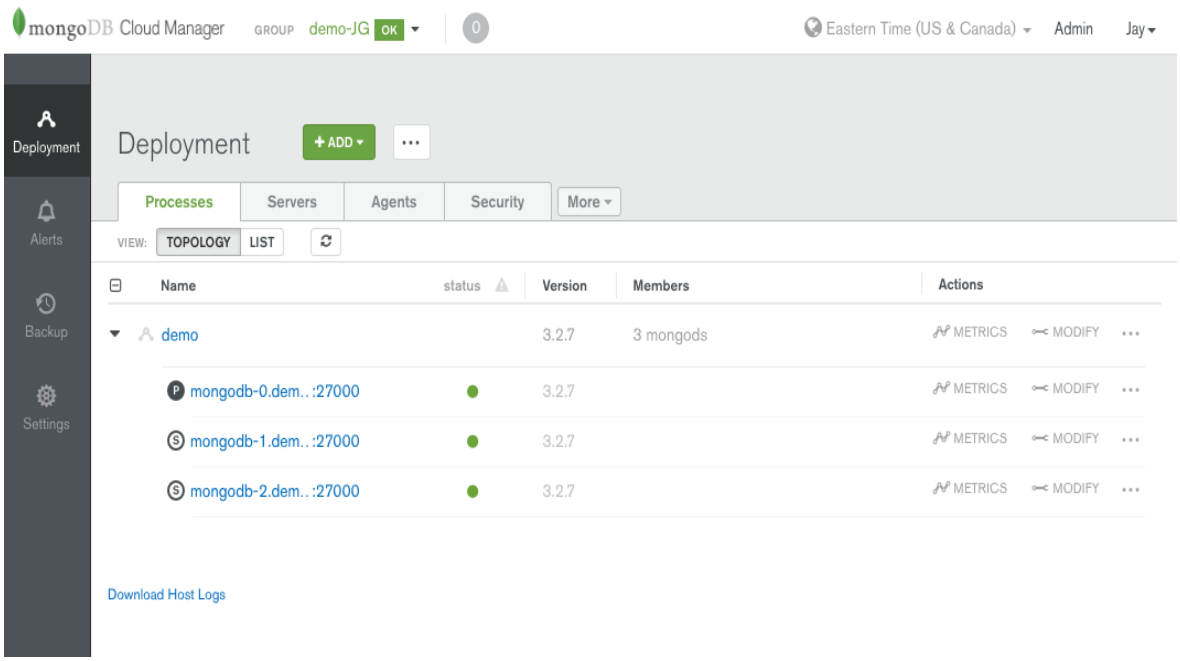


Рисунок 3.4 —Інтерфейс Atlas MongoDB

Важливо відзначити, що механізм створення резервних точок відновлення використовується для підвищення надійності системи сховищ даних, оскільки всі дані є одночасно декількома копіями і не можуть бути легко втрачені, оскільки кожна така копія є фізично незалежною. Повна втрата даних може бути досягнута тільки тоді, коли всі потужні сервери не працюють одночасно.

3.4 Python Telegram Bot Api

Модуль, який є обгорткою для офіційної версії Telegram API, що дозволяє виконувати операції за допомогою ботів. В зв'зці з застосунком Telegram є можливість створення гнучких систем для отримання інформації від користувачів у знайомому інтерфейсі, в якому користувач звик проводити свій час.

Переваги, які надає використання Telegram-бота в якості клієнта:

1. знайомий для користувача інтерфейс, до якого не потрібно звикати чи пристосовуватись;

2. можливість отримувати велику кількість даних, завдяки існування груп користувачі, що може знадобитись при розробці рекомендаційної моделі;

3. потрібно визначати лише логіку, вся взаємодія відбувається на стороні Telegram.

Для того, щоб зберігати дані з клієнта, в якості кешу, дані зберігаються до файлу у форматі JSON. У перспективі для збереження неструктурованих даних, повідомлень користувачів Telegram та інших видів користувацьких даних краще перейти на MongoDB, яка забезпечує більше функціональних можливостей.

3.5 Gensim

Бібліотека Gensim (рисунок 4.5) з відкритим сирцевим кодом, яка дозволяє використовувати інструменти для тематичного моделювання наборів текстів, що є доцільним в контексті описаної задачі.

Рисунок 3.5 — Gensim

Gensim починався як набір різних сценаріїв Python для Чеської цифровий математичної бібліотеки dml в 2008 році, де був створений короткий список статей, найбільш близьких до даної статті (gensim = "генерувати схожі"). У публікації LREC за 2010 рік описані початкові проектні рішення Gensim: ясність, ефективність і масштабованість.

Пізніші версії gensim значно підвищили цю ефективність і масштабованість. До теперішнього часу, Gensim є найбільш надійним, ефективним і простим програмним забезпеченням для реалізації неконтрольованого семантичного моделювання зі звичайного тексту.

3.6 Висновки до розділу

Було розглянуто всі програмні засоби, що було використано в ході розробки програмного продукту з їх детальним описанням. Проведено аналіз аналогів та визначення кращого варіанту в даній предметній області.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для забезпечення універсальності та можливості для масштабування було прийняте рішення реалізувати сервіс на базі веб-застосунку з виділеним API та базовим клієнтом використовуючи Telegram-Bot API, що дозволяє в повній мірі вирішити поставлену проблему, з використанням існуючих ресурсів для зосередження уваги на реалізації бізнес-логіки застосунку.

4.1 Загальна структура

В центрі системи знаходиться серверний модуль реалізований з використанням модуля Flask в якості основи для сервера. В якості обгортки для базового сервера працює модуль Flask-Restful, де визначені базові структри, як ресурси, поля та класи для отримання атрибутів із запитів, робить процес реалізації API ефективнішим та робить простішим подальше тестування.

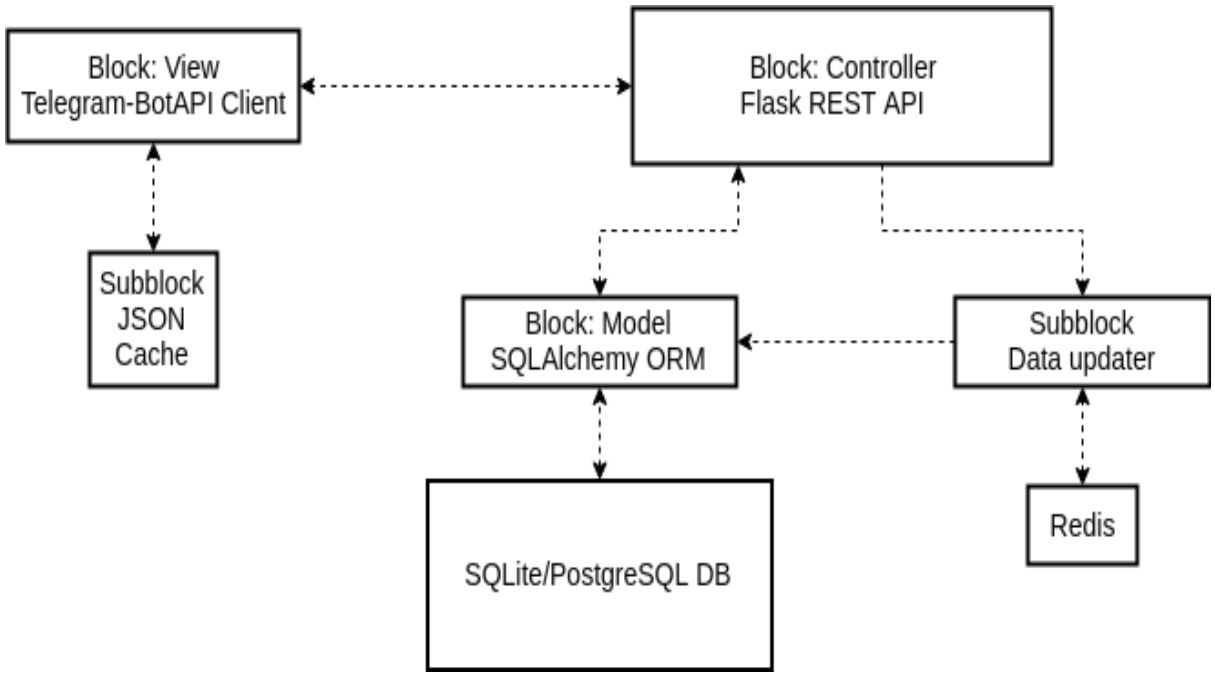


Рисунок 4.1 — Структура проекту з виділеними основними модулями

Сервер оперує даними, які приходять від клієнта, а також виконує запити до бази даних, для спрощення реалізації взаємодії з БД використано ORM, що дозволяє економити час для покращення бізнес-логіки чи тестування наявного функціоналу.

Також для того, щоб дані, які доступні користувачу були актуальними використовується підмодуль на базі фреймворку Celery. Визначено кілька типів завдань, які працюють з інтервалами в тиждень і виконують періодичне оновлення даних. Черга завдань зберігається в Redis, це нереляційна база даних, що зберігається в оперативній пам’яті. Вона є досить швидкою та надійною, гарантуючи, що всі завдання збережуться при аварійному завершенні роботи системи.

Завдання на періодичне оновлення даних, використовують RozkladKPI API і містять інформацію про навчальні групи, предмети та графік занять. Це корисно, адже лише якщо користувач вкаже свою групу в додатку, можна отримати інформацію про напрямок його навчання, викладачів і можливо навіть оцінювати відвідування за допомогою стандартного функціоналу доступного в Telegram. Також це дозволяє розділяти запитання на категорії і в подальшому може спростити пошук та рекомендації схожих або майбутніх запитань.

4.2 Архітектура серверної частини

Шаблон проектування — це архітектура, рішення яке описує яким чином вирішуються задачі, які часто зустрічаються при розробці програмних систем чи додатків.

Для реалізації серверу було використано фреймворк, який є реалізацією шаблону проектування MVC.

MVC (Model-View-Controller) — це шаблон проектування, головною ідеєю якого є відділити логіку застосунку від представлення (рисунок 4.2). Принципом MVC є розділення програмної реалізації системи на три головні компоненти: M — Model (Модель), V — View (Представлення), C — Controller (Контролер), таким

чином, що редагування будь-якого компонента може відбуватися незалежно.

Модель — містить знання про предметну область, дані та правила доступу до цих даних, але нічого не знає про контролери та представлення. У моделі відбувається бізнес-логіка роботи застосунку. Модель надає контролеру дані які запрошує користувач, або інша система.

Представлення — надає можливість по-різному відображати дані отримані будь-яким способом від моделі, може містити в собі логіку. Представлення — це кінцевий інтерфейс, з яким взаємодіє користувач. Користувач може передавати дані через представлення.

Контролер — реалізує взаємодію між моделлю і представленням. У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов’язаних дій в окремий клас.

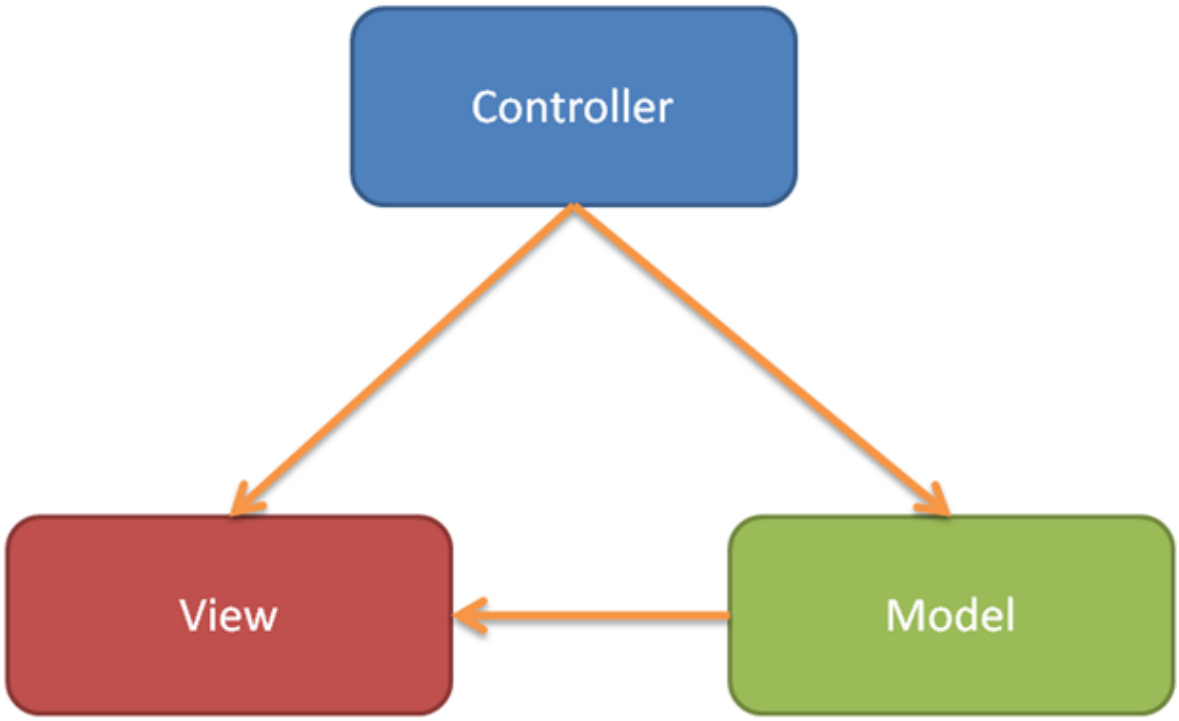


Рисунок 4.2 — Схема роботи MVC шаблону

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (представлення). За рахунок такого поділу підвищується

можливість повторного використання та гнучкість системи. Також для великих проектів, застосування шаблону MVC або його похідних дозволяє застосувати тестування різноманітних компонентів програмної системи.

4.3 Архітектура клієнтської частини

Як було описано в попередніх параграфах, в якості клієнта використовується Telegram-бот реалізований на базі популярної бібліотеки python-telegram-bot. Використання готової платформи для створення клієнта є дуже вигідним кроком, адже користувачам не потрібно виходити з власної зони комфорту при взаємодії з подібним застосунком. Також це можливість дізнатись більше про власних користувачів, щоб покращувати наявний функціонал, адже в цьому випадку набагато простіше збирати зворотній зв’язок.

В загальному для того, щоб створити бота в Telegram є кілька десятків методів, які дозволяють будувати інтерфейс для взаємодії з користувачем. Було використано методи, які дозволяють контролювати надходження певної команди до бесіди з користувачем і вчасно реагувати потрібним чином. Є можливість задавати шаблони, які будуть перехоплювати і обробляти повідомлення, це дозволяє будувати ланцюжки функцій для послідовного виклику.

Кожен ланцюжок введення користувацьких даних у реалізованому клієнті побудовано на базі кінцевих автоматів. Це дозволяє чітко визначати стани, в яких перебуває користувач для обмеження дій, які користувач може виконати в поточний момент.

На рисунку 5.3 продемонстровано послідовність станів, які проходить користувач при введенні даних про свою навчальну групу. Для початку він просто вводить дані, далі виконується така послідовність пошук в кеші - пошук в базі даних - виконання пошукового запиту через Rozklad KPI API, слідуючим кроком буде затвердження вибору і подальший вихід.

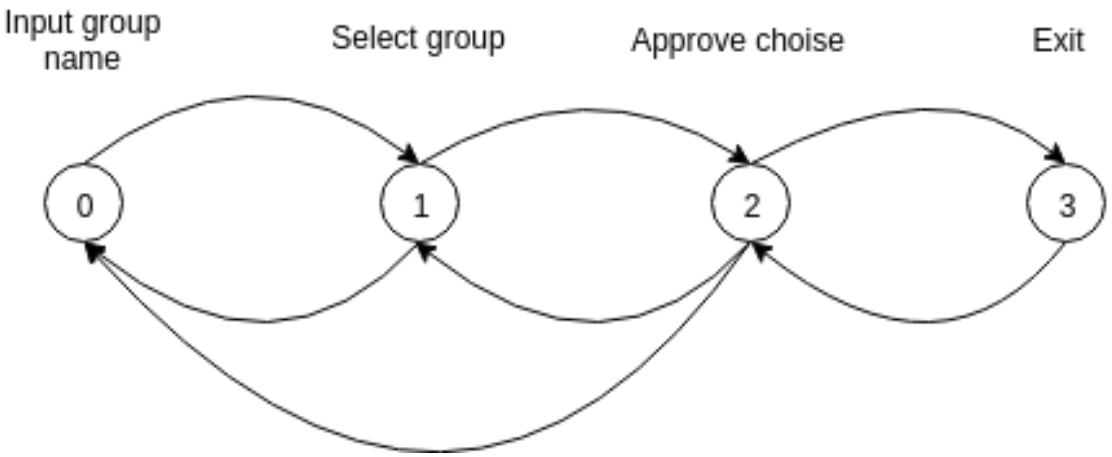


Рисунок 4.3 — Приклад кінцевого автомату для вибору групи

Кеш для запитів, що вже виконувались побудований поверх простого модуля tinyDB, що використовує JSON файли в якості сховище. Це забезпечує швидше виконання запитів, які використовуються часто, тобто загалом оптимізує швидкість роботи клієнта.

4.4 Тестова рекомендаційна системи з існуючим набором даних

Рекомендаційна система є важливим елементом для сервісів, які містять в собі запитання, адже це допомагає знаходити відповідь набагато швидше.

Було обрано англомовний набір даних (рисунок 5.4) для того, щоб почати розробку вже на готових даних та продовжити її після впровадження.

Дані, було обрано за схожістю спрямування з ресурсу CareerVillage. В свою чергу вони розділені по файлах:

1. comments.csv: Коментарі можуть бути зроблені на відповіді або запитання. Ми посилаємося на будь-який коментар, який викладено як "батько" цього коментаря. Коментарі можуть розміщуватися будь-яким користувачем.
2. answers.csv: Відповіді є те, про що це все! Відповіді викладаються у відповідь на запитання. Відповіді можуть публікувати лише користувачі, зареєстровані як професіонали. Однак, якщо хтось змінив свій тип реєстрації після приєднання, вони можуть з'явитися як автор відповіді, навіть якщо вони

більше не є професіоналом.

3. emails.csv: Кожне повідомлення електронної пошти відповідає одному електронному листу певному одержувачу. Рівень frequency_level відноситься до типу шаблону електронної пошти, який включає негайну електронну пошту, надіслану відразу після запиту, щоденних збірок та щотижневих збірок.
4. group_memberships.csv: Будь-який користувач може приєднатися до будь-якої групи. До цих пір є лише кілька груп.
5. groups.csv: Кожна група має тип. З міркувань конфіденційності ми повинні залишити імена груп.
6. matches.csv: Кожен рядок повідомляє, які запитання були включені в електронні листи. Якщо електронна пошта містить лише одне запитання, ідентифікатор цього листа з'явиться лише один раз. Якщо електронний лист містить 10 запитань, цей ідентифікатор електронної пошти буде відображатися тут 10 разів.
7. professionals.csv: Ми називаємо наших волонтерів "Професіоналами", але ми можемо також назвати їх супергероями. Вони - дорослі, які добровільно віддають свій час, щоб відповісти на запитання на сайті.
8. questions.csv: Питання викладаються студентами. Іноді вони дуже просунуті. Іноді вони просто починають. Все це справедлива гра, якщо вона стосується майбутнього професійного успіху студента.
9. school_memberships.csv: Так само, як group_memberships, але для шкіл замість.
10. students.csv: Студенти, як правило, коливаються у віці від 14 до 24 років.
11. tag_questions.csv: Кожне питання може бути хештег. Ми відстежуємо сполучення хештег-на-запитання і поміщаємо їх у цей файл.
12. tag_users.csv: Користувачі будь-якого типу можуть слідувати хештегу. Це показує, які хеш-теги слід кожен користувач.
13. tags.csv: Кожен тег отримує ім'я.
14. question_scores.csv: "Серця" за кожне запитання.
15. answer_scores.csv: "Серця" за кожну відповідь.

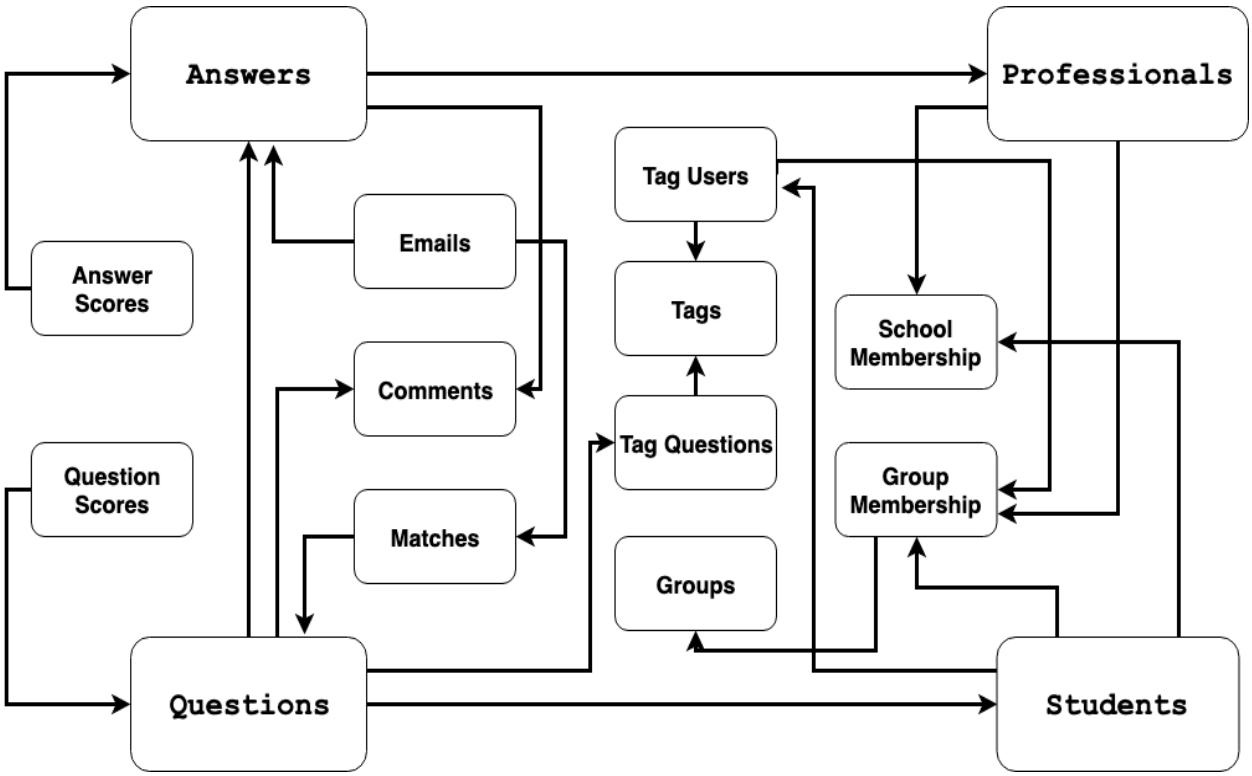


Рисунок 4.4 — Структура набору даних

Для розподілу запитань за темами для подальшого пошуку подібних використовується Латентне розміщення Діріхле (LDA), яке застосовується в машинному навчанні та інформаційному пошуку для визначення тем до яких належить текст.

Це дає можливість побачити з якою вірогідністю розподілені запитання в нашому наборі даних. Для того, щоб отримати результат запропоновано такий алгоритм:

Використовуючи засоби обробки природної мови потрібно підготувати набір даних який використовується.

Визначити частини мови для того, щоб відфільтрувати зайві слова.

Визначити частоту та обернену частоту слів у корпусі даних.

Тренування LDA моделі.

Тестування на новому наборі даних.

Після проведеної обробки ми можемо отримати так звані Word clouds (рисунок 5.5), які дозволяють в графічному вигляді побачити слова, які найчастіше зустрічаються в запитаннях. Більший розмір тексту означає більшу частоту слова.

Модель LDA створюється за допомогою бібліотеки Gensim. Існує безліч параметрів, які можуть бути встановлені і впливати на тренування модель. Це неконтрольована модель і нелегко вибрати правильні параметри. Було протестоване рішення з різними параметрами.

Параметри Gensim Dictionary Filter призначені для попередньої обробки, фільтрації лексем в словнику по їх частоті. Параметри LDA призначені для навчання моделі. Найцікавішим є num_topics, який визначає, скільки тим ми хочемо мати.

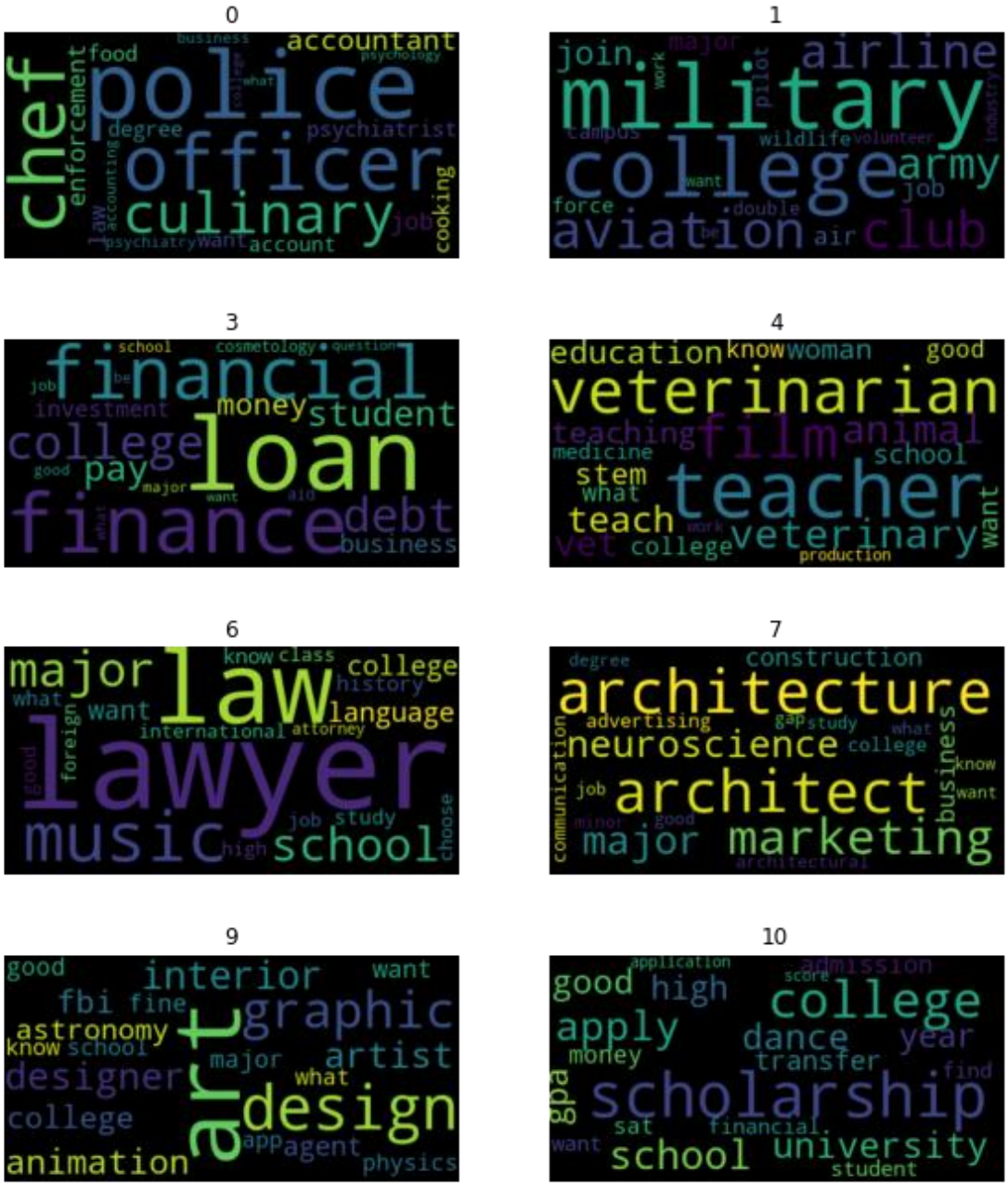


Рисунок 5.5 — Word clouds з словами, які зустрічаються

Проте в тестовому варіанті рекомендаційна система не реалізована до кінця, адже це не було головною метою дослідження та розробки продукту.

4.5 Висновок до розділу

У розділі було подано інформацію про архітерну складову системи та розглянуто особливості кожного процесу більш детально з наглядною демонстрацією схем, які відносяться до компонентів системи.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Розроблено програмний комплекс розроблений з використанням веб-технологій і тому його можна використовувати на будь-якій платформі, яка підтримують актуальні веб-стандарти. Додаткових встановлень, окрім авторизації не вимагає, система легка та адаптивна під різні пристрої.6.1 Опис функціональності системи

Клієнт повинен якимось чином взаємодіяти з системою, це можна зобразити за допомогою Use case діаграми. Графічне зображення акторів та доступних для них дій, дає можливість наглядно зрозуміти, яким чином повинна працювати система.

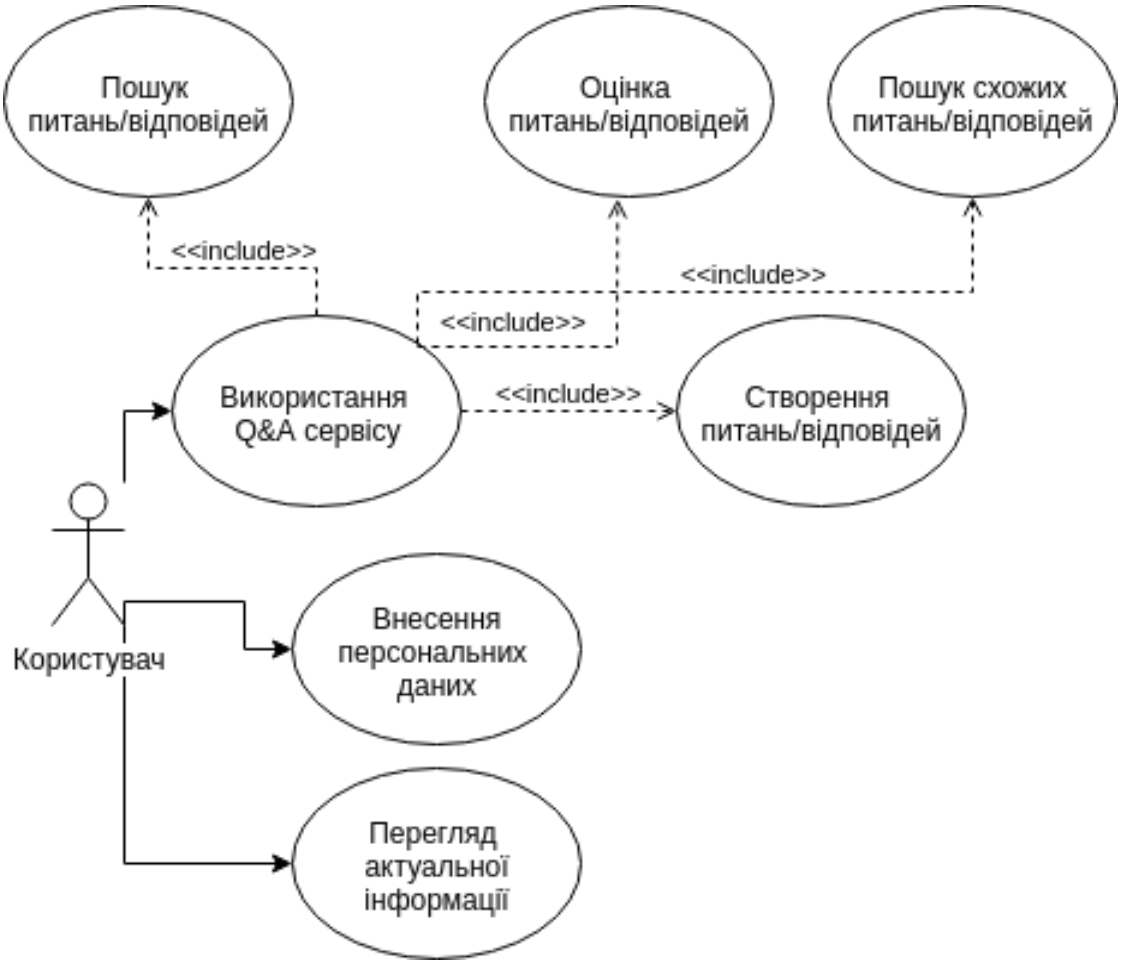


Рисунок 5.1 — Use case діаграма для користувача системи

На рисунку 5.1 зображено можливий функціонал для користувача, виключаючи реєстрацію та аутентифікацію, як речі, які не відносяться до задач предметної області, що досліджується. Загалом описано всі можливі види взаємодії користувача з системою, хоча , функції якими забезпечує користувача система можуть бути розширені.

5.1 Інсталяція та системні вимоги

Система дозволяє використовувати API, тобто теоретично можна отримати доступ з будь-якого пристрою підключеного до мережі з можливістю відправляти запити. Проте клієнт реалізований для Telegram, який доступний на мобільних пристроях з Android та iOS, також додаток доступний для всіх поширених десктопних систем Windows, Linux та macOS. Є можливість використовувати і веб

версію додатку, тобто доступ можливий з будь-якого пристрою з веб-браузером.

Інсталяція не потрібна у випадку використання веб додатку. Потрібно лише мати обліковий запис в Telegram.

5.2 Сценарій роботи користувача з системою

Якщо користувач має зареєстрований аккаунт в Telegram, використовуючи пошук можна знайти бота за посиланням @APEPSAssistantBot, після чого можна побачити повідомлення, що зображене на Рисунку 5.2

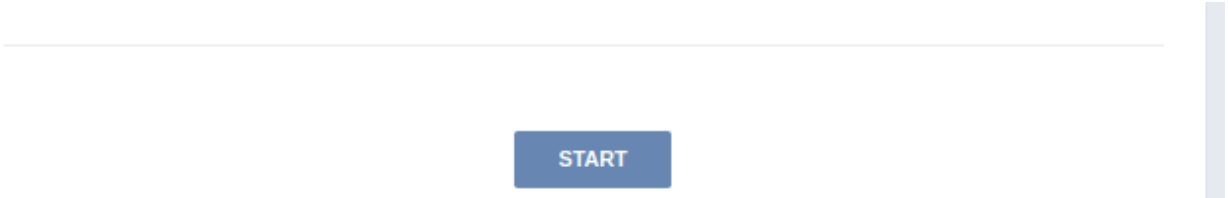


Рисунок 5.2 — Початок роботи

Після старту буде виведено повідомлення та контекстне меню для вибору потрібної функції. Для користувача, який вперше почав бесіду з ботом або для того, хто ще не обрав навчальну групу, буде запропоновано зробити свій вибір.

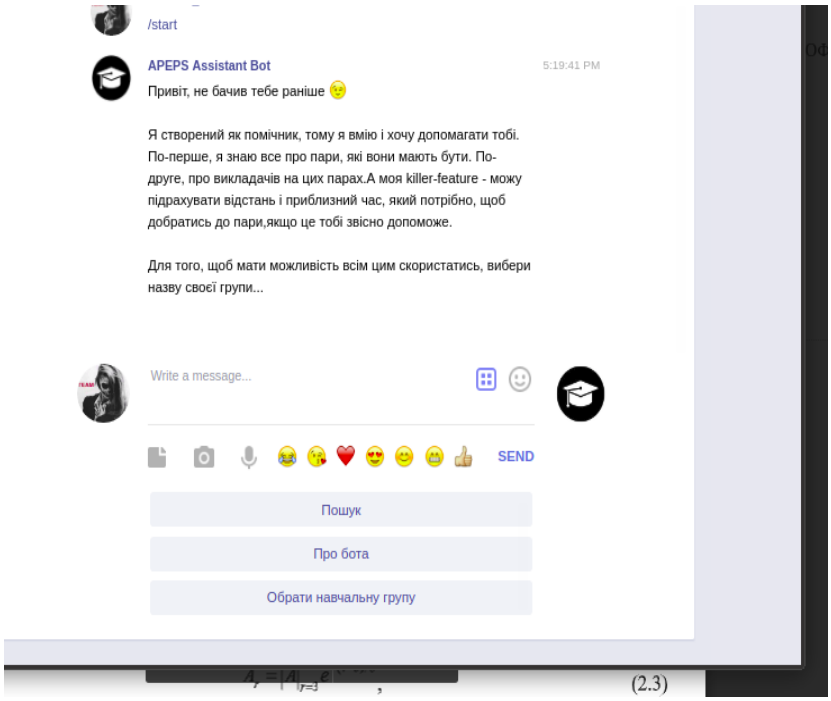
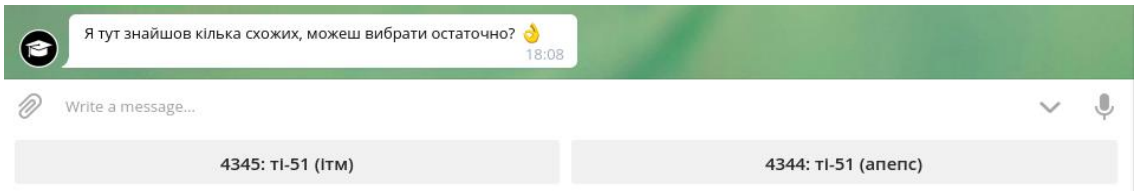


Рисунок 5.3 - Меню Telegram бота

Для того, щоб мати можливість повністю використовувати функціонал боту, потрібно обрати навчальну групу. Для цього потрібно натиснути на “Обрати навчальну групу” в меню бота. В попередніх розділах було описано процес вибору навчальної групи з боку серверної сторони, те як веде себе клієнт продемонстровано на рисунку 5.4.



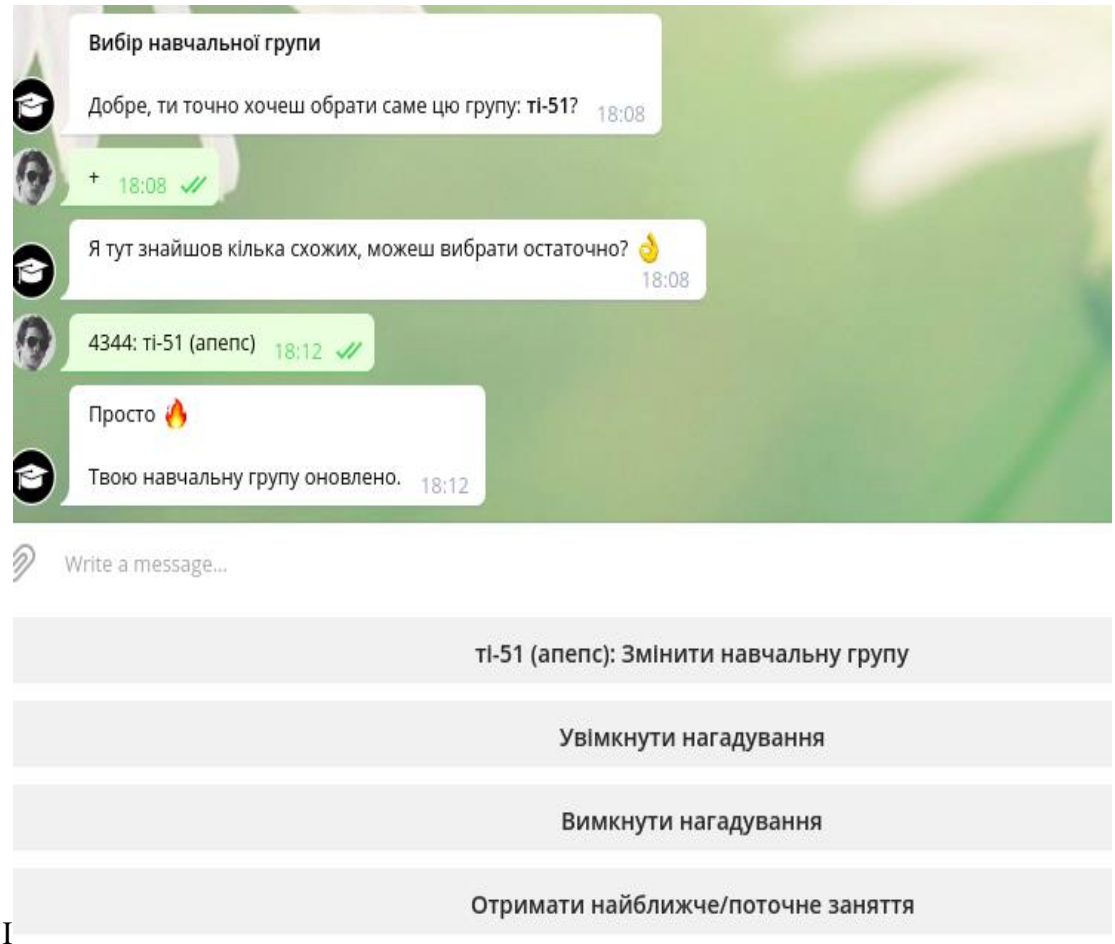


Рисунок 5.4 — Діалог з Telegram Bot

На карті є елементи керування для створення полігонів за допомогою маніпуляцій з комп’ютерною мишею, вони знаходяться у правому верхньому куту мапи (рисунок 5.3). Також там знахоться іконки редагування створених полів та видалення, в разу необхідності.

В розділі описано вимоги для запуску програмного забезпечення, які є мінімальними, коротка інструкція по інтерфейсу веб-застосунку з детальними скріншотами, що ілюструють систему.

ВИСНОВКИ

У ході виконання диплома розроблена система, яка вже вирішує базові проблеми студентів. Перш за все вирішується проблема пошуку потрібної інформації, відповіді отримуються в режимі реального часу, що забезпечує належний рівень актуальності інформації.

Система включає модуль для доступу до ресурсів університету для збору інформації про навчальний процес, а також сервіс, який забезпечує програмний інтерфейс для зовнішнього доступу.

Дана система відповідає списку поставлених задач:

1. Створено модуль для взаємодії з API Rozklad KPI.
2. Написано сервіс з можливим доступом з зовнішніх ресурсів чи додатків.
3. Надано можливість взаємодіяти з іншими користувачами за допомогою запитань та відповідей.
4. Протестована робота сервісу в зв’язці з Telegram-ботом.
5. Протестовані можливі рішення для впровадження рекомендаційної системи.

Кінцева платформа готова до розширення функціоналу за рахунок мікросервісної архітектури, що зменшує рівень зчепленості різних модулів системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методика використання інформаційно-комунікаційних технологій у навчальному процесі. Ч.4. Проектування методів управління навчальною діяльністю:начальний посібник / Стариченко Б.Е., Коротаєва Е.В., Сардак Л.В., Єгоров А.Н. , Під ред. Стариченко Б.Е. Єкатеринбург: 2013. 141 с.

2. Методика викладання економічних дисциплін: навчально-методичний комплекс /Хвесеня Н. П.,Під ред. Н. П. Хвесеня, М. В. Саковича. Мінськ: БДУ, 2006. 116 с.

3. Яркіна Т.Н. Педагогіка середньої професійної освіти: курс лекцій. Томськ:ТДПУ, 2009. 272с.

4. Гладких Б. А. Вибір шкали оцінки знань у вузі в контексті Болонського процесу // Проблеми управління в соціальних системах. 2011. No5. С.98-118.

5. Челышкова М.Б. Теорія і практика конструювання педагогічних тестів: навчальний посібник М.: Логос, 2002.432 с

6. Flask (A Python Microframework) [Electronic resource]. — Access mode: http://flask.pocoo.org/.

7. Flask-RESTful — Flask-RESTful 0.3.7 documentation [Електронний ресурс] – Режим доступу до ресурсу: https://flask-restful.readthedocs.io.

8. Telegram Bot API - Telegram APIs [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/bots/api.

9. PyCharm: the Python IDE for Professional Developers by JetBrains [Електронний ресурс] — Режим доступу: https://www.jetbrains.com/pycharm

10. Spring Boot [Електронний ресурс] — Режим доступу: https://spring.io/projects/spring-boot/

11. gensim: Topic modelling for humans - Radim Řehůřek [Електронний ресурс] – Режим доступу до ресурсу: https://radimrehurek.com/gensim/.

12. Riel A. J. Object-oriented design heuristics / Arthur J. Riel., 2019. – 231 с.

13. Bologna Working Group on Qualifications Frameworks (2005) A Framework for Qualifications of the European Higher Education Area. [Електронний ресурс] – Режим доступу до ресурсу: http://www.bologna-bergen2005.no/Docs/00-Main_doc/050218_QF_EHEA.pdf

14. ECTS Users Guide. Office for Official Publications of the European Communi-ties: Luxembourg, 2009. [Електронний ресурс] – Режим доступу до ресурсу: http://ec.europa.eu/education/lifelong-learning-policy/doc/ects/guide_en.pdf.

15. European Credit Transfer System. ECTS User's Guide / Luxemburg: Office for Official Publications of the European Communities. 1995. 302 p.

ДОДАТОК 1

Розробка серверної частини для системи обліку успішності студента

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51179_19Б

Аркушів 2

Київ – 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ51179_19Б	Записка.doc	Пояснювальна записка
Компоненти		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ51179_19Б	common/errors.js common/decorators.js common/__init__.py resources/questions.py resources/users.py resources/auth.py resources/ __init__.py main.py	API сервіс системи
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТВ51179_19Б	core.py errors.py	Модулі, які реалізують доступ до Rozklad KPI

ДОДАТОК 2

Модуль для аутентифікації користувача та взаємодії з основними сутностями

Лістинг програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51179_19Б 12-1

Аркушів 5

Київ – 2019

```
from flask import request, Blueprint, g
from flask_jwt_extended import (
    jwt_required, create_access_token, get_jwt_identity,
    create_refresh_token, jwt_refresh_token_required, get_raw_jwt)
from flask_restful import Resource, abort, marshal, marshal_with, Api
from werkzeug.security import generate_password_hash, check_password_hash

from service_app.common.errors import IncorrectRequestBodyError, ExistingUserError
from service_app.resources.users import user_fields


bp = Blueprint("auth", __name__, url_prefix="/auth")
api = Api(bp, prefix="/api/v1")


class Register(Resource):
    @marshal_with(user_fields)
    def post(self):
        if not isinstance(request.json, dict):
            abort(IncorrectRequestBodyError)

        username = request.json.get("username", None)
        password = request.json.get("password", None)

        if connector.db.users.find({"username": username}).count():
            abort(ExistingUserError.code, message=ExistingUserError.message)

        user = {key: request.json.get(key, None) for key in user_fields.keys()}

        if password is None:
            user["password"] = generate_password_hash(user["username"] + user["id"])
        else:
            user["password"] = generate_password_hash(user["password"])
        user["access_token"] = create_access_token(identity=user["username"])
        user["refresh_token"] = create_refresh_token(identity=user["username"])
        connector.db.users.insert(marshal(user, user_fields))
        return user


class Login(Resource):
    def post(self):
        data = request.json
        username = data['username']
        try:
            current_user = connector.db.users.find({"username": username})[0]
        except IndexError as e:
            return {
                "error": {
                    "status": 404,
                    "message": "Unexpected attribute value in field 'username'",
                }
            }
```

```
        if current_user["password"] and (
                check_password_hash(current_user["password"], data["password"]) or
                check_password_hash(current_user["password"],
generate_password_hash(data["id"]+data["username"]))
        ):
            g.current_user = current_user

            return {
                'access_token': create_access_token(identity=username),
                'refresh_token': create_refresh_token(identity=username),
            }
        else:
            return {'message': 'Wrong credentials'}


class LogoutAccess(Resource):
    @jwt_required
    def post(self):
        jti = get_raw_jwt()['jti']
        try:
            connector.db.tokens.insert({"jti": jti})
            return {'message': 'Access token has been revoked'}
        except:
            abort(500, message="Something went wrong with your token.")


class LogoutRefresh(Resource):
    @jwt_refresh_token_required
    def post(self):
        jti = get_raw_jwt()['jti']
        try:
            connector.db.tokens.insert({"jti": jti})
            return {'message': 'Refresh token has been revoked'}
        except:
            abort(500, message="Something went wrong with your token.")


class TokenRefresh(Resource):
    @jwt_refresh_token_required
    def post(self):
        current_user = get_jwt_identity()
        access_token = create_access_token(identity = current_user)
        return {'access_token': access_token}


api.add_resource(Register, "/register")
api.add_resource(Login, "/login")

api.add_resource(TokenRefresh, "/get_token")

api.add_resource(LogoutAccess, "/access_token_revoke")
```

```
api.add_resource(LogoutRefresh, "/refresh_token_revoke")


from service_app.app import connector

import re
from datetime import datetime

from bson.objectid import ObjectId
from flask_restful import Resource, fields, marshal_with, reqparse, abort, marshal
from pymongo import MongoClient

client = MongoClient("")


question_parser = reqparse.RequestParser()
question_parser.add_argument(
    "text", required=True, help="Body of the question that include simple text. Required field."
)
question_parser.add_argument(
    "tags", help="One or more tags that describe the question body, use space as a delimiter "
                 "and dash '-' if tag include more than one word."
)


class AbstractCrudTextObject(object):
    base_fields: dict = {
        "_id": fields.String(),
        "text": fields.String(),
        "tags": fields.List(fields.String()),
        "rating": fields.Integer(default=0),
        "created_at": fields.DateTime(),
    }

    fields: dict
    collection_name: str

    def init(self, db_client):
        self.db_client = db_client

    @staticmethod
    def extract_tags_from_text(text):
        return re.findall("#[A-Za-zA-zА-Яа-я-]+", text)

    @classmethod
    def get_collection_from_db(cls, client):
        return getattr(client.get_database("university"), cls.collection_name)

    @classmethod
    def get_fields(cls):
        return dict(cls.base_fields, **cls.fields)
```

```
    @classmethod
    def find_one_by_id(cls, client, _id):
        return cls.get_collection_from_db(client).find_one({"_id": ObjectId(_id)})


class AnswerModel(AbstractCrudTextObject):
    collection_name = "answers"
    fields = dict()


class QuestionModel(AbstractCrudTextObject):
    collection_name = "questions"
    fields = {
        "answers": fields.Nested(AnswerModel.fields),
    }


class QuestionList(Resource):

    @marshal_with(QuestionModel.get_fields())
    def get(self):
        return list(QuestionModel.get_collection_from_db(client).find())

    def post(self):
        args = question_parser.parse_args()

        try:
            question_collection = QuestionModel.get_collection_from_db(client)
            question = marshal(args, QuestionModel.get_fields())
            del question["_id"]

            question["created_at"] = datetime.now()
            question["tags"] = QuestionModel.extract_tags_from_text(question["text"])

            if not question_collection.find_one({"text": question["text"]}):
                data = question_collection.insert_one(question)
                print(f"Inserted {data}")
            else:
                abort(400, message="Entity with similar body already exists!")

            return {
                "question": marshal(question, QuestionModel.get_fields()),
                "status_code": 201,
                "message": "New question created successful!"
            }
        except ValueError as e:
            abort(400, message="Bad post request. Text field is required!", error=str(e))


feedback_parser = reqparse.RequestParser()
```

```
feedback_parser.add_argument("vote", help="Include user ID who give this vote.")
feedback_parser.add_argument("tags")
feedback_parser.add_argument("answer")


class Question(Resource):

    @marshal_with(QuestionModel.get_fields())
    def get(self, question_id):
        question = QuestionModel.find_one_by_id(client, question_id)
        if question:
            return question
        else:
            abort(404, message="Question entity was not found!")

    @marshal_with(QuestionModel.get_fields())
    def post(self, question_id):
        args = feedback_parser.parse_args()

        if "vote" in args:
            QuestionModel \
                .get_collection_from_db(client) \
                .update_one({"_id": ObjectId(question_id)},
                            {"$inc": {"rating": 1}})
            return QuestionModel.find_one_by_id(client, question_id)
```

ДОДАТОК 3

Модуль для аутентифікації користувача та взаємодії з основними сутностями

Опис програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51179_19Б 12-2

Аркушів 8

Київ – 2019

АНОТАЦІЯ

Додаток надає можливість для авторизації та взаємодії з іншими користувачами системи забезпечуючи постійний обмін корисною інформацією. Також забезпечено підтримку актуальності навчальної інформації.

За допомогою методів реалізованих у модулі виконується збереження, попередня обробка та оцінка даних, які надходять від користувачів.

Весь програмний код був написаний за допомогою мови програмування Python.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Програмний код міститься в кількох файлах, що формують модуль. Він включає методи для збереження та обробки користувацької інформації, а саме запитань та відповідей, що спрямовано на покращення ефективності навчання. Також забезпечено доступ до актуальної навчальної інформації у модулі для доступу до ресурсу Rozklad KPI.

Необхідні дані очікує на вхід сервіс написаний на Python, він також виконує і подальшу обробку вхідної інформації, через забезпечено прямий доступ до бази даних MongoDB.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Сервіс реалізує наступний функціонал:

— приведення строкових даних про координати до географічного типу з даними про широту і довготу;

— доступ до актуальної навчальної інформації пов’язаної з розкладом;

— можливість задавати запитання та відповідати на існуючі;

— оцінка існуючих запитань та відповідей;

— формування рейтингу на основі оцінок.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Даний модуль реалізовує базові класи для доступу до потрібнних сутностей, а саме користувацьких токенів доступу та самих запитань. Весь функціонал доступний в слідуючих класах:

1) Register;
2) Login;
3) Logout;
4) LogoutRefresh;
5) AbstractCrudTextObject;
6) Question;
7) QuestionModel.

ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУВАЛИСЯ

Програмний код модулю було написано у середовищі розробки PyCharm, що спростило процес зневадження та тестування.

Прогрмана реалізація написана за допомогою Python, кількох клієнтських бібліотек та фреймворку Flask.

Для реалізації тематичного моделювання та інших моделей для роботи з такого роду текстовою інформацією застосовано відкриту бібліотеку Gensim.

-8-

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними є:

— Необроблений текст запитань, що містяться в JSON-файлах.

Вихідними даними є:

— агреговані відповіді на поставлені запитання;

— рейтинг для запитань і відповідей.